(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 111831272 A
(43)申请公布日 2020.10.27

(21)申请号 201910299903.7

(22)申请日 2019.04.15

(71)申请人 阿里巴巴集团控股有限公司
地址 英属开曼群岛大开曼资本大厦一座四层847号邮箱

(72)发明人 李剑 陈超 刘天骐 李晓昱 王喆

(74)专利代理机构 上海知锦知识产权代理事务所(特殊普通合伙) 31327
代理人 潘彦君 李丽

(51)Int.Cl.
G06F 8/34(2018.01)

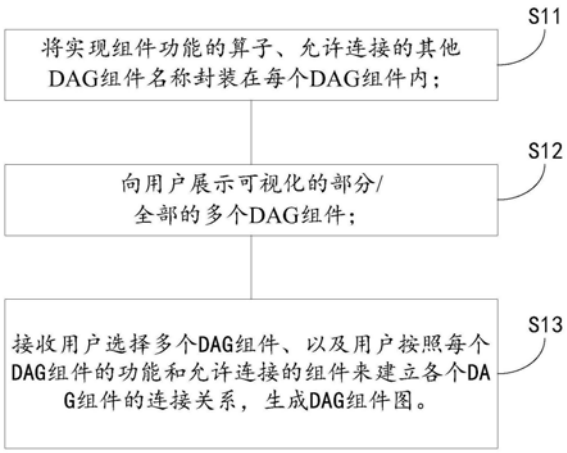
权利要求书2页 说明书7页 附图5页

(54)发明名称

一种采用图形化的开发的方法、介质、设备和装置

(57)摘要

本发明提供一种采用图形化的开发的方法、介质、设备和装置。方法包括:将实现组件功能的算子、允许连接的其他DAG组件名称封装在每个DAG组件内;向用户展示可视化的多个DAG组件;接收用户选择多个DAG组件、以及按照每个DAG组件的功能和允许连接的组件来建立各个DAG组件的连接关系,生成DAG组件图。通过上述的实施例,将用代码实现相应组件功能的算子,封装成DAG组件,以图形化的方式显示软件开发过程,简化了用户的开发流程,用户不比关注每条具体指令的表述,以及语法关系,这样的开发模式将代码的算子抽象、封装成可视化、可配置化的前端组件,更加直观,便于用户理解。



CN 111831272 A

1.一种采用图形化的开发的方法,其特征在于,包括:

将实现组件功能的算子、允许连接的其他DAG组件名称封装在每个DAG组件内;

向用户展示可视化的多个DAG组件;

接收用户选择多个DAG组件、以及按照每个DAG组件的功能和允许连接的组件来建立各个DAG组件的连接关系,生成DAG组件图。

2.根据权利要求1所述的方法,其特征在于,所述封装的内容还包括:一个或多个的参数或属性配置项;

所述接收用户选择的所述DAG组件的过程,还包括:

对所述用户选择的DAG组件,显示参数或属性配置面板;

接收用户输入的配置的参数或属性数据,并实时校验是否正确。

3.根据权利要求1所述的方法,其特征在于,还包括:

在所述连接关系上,每增加一个DAG组件,所述连接关系发生变化后,对新增加的DAG组件的前路的正确性进行验算;所述前路包括连接关系上的各个DAG组件及形成路径;

或,点击任一个DAG组件,对其前路的DAG组件的路径正确性进行验算。

4.根据权利要求3所述的方法,其特征在于,所述验算过程包括:检查每个DAG组件的一个或多个的输入、输出的字段列表和类型的正确性。

5.根据权利要求1所述的方法,其特征在于,还包括:在已经建立的多个DAG组件形成的连接关系上,框选多个DAG组件,作为一个可调用模板进行保存。

6.根据权利要求1所述的方法,其特征在于,所述生成DAG组件图之后,还包括:分析所述DAG组件图的代码关系。

7.根据权利要求6所述的方法,其特征在于,所述分析的过程包括:

将所述各个DAG组件及连线形成的所述连接关系,分解为数据结构;

将分解后的数据结构,按照语句的构成,拆分成多个逻辑算子;

所述拆分后的多个逻辑算子,生成相应的语句代码。

8.根据权利要求7所述的方法,其特征在于,所述生成相依的语句代码后,还包括:接收切换指令,对所述分析过程,执行逆操作,形成所述DAG组件图。

9.一种计算机可读存储介质,其特征在于,存储有计算机程序,该程序被处理器执行时,实现权利要求1～8任一项所述的方法。

10.一种计算机设备,其特征在于,安装有权利要求9所述的计算机存储介质。

11.一种采用图形化的开发的装置,其特征在于,包括:

封装模块,用于将实现组件功能的算子、允许连接的其他DAG组件名称封装在每个DAG组件内;

展示模块,用于向用户展示可视化的多个DAG组件;

图形模块,用于接收用户选择多个DAG组件、以及按照每个DAG组件的功能和允许连接的组件来建立各个DAG组件的连接关系,生成DAG组件图。

12.根据权利要求11所述的装置,其特征在于,所述封装的内容还包括:一个或多个的参数或属性配置项;

所述图形模块,还连接有参数校验模块,用于对所述用户选择的DAG组件,显示参数或属性配置面板;

接收用户输入的配置的参数或属性数据,并实时校验是否正确。

13.根据权利要求12所述的装置,其特征在于,所述图形模块,还连接有验算模块,用于在所述连接关系上,每增加一个DAG组件,所述连接关系发生变化后,对新增加的DAG组件的前路的正确性进行验算;所述前路包括连接关系上的各个DAG组件及形成路径;

或,点击任一个DAG组件,对其前路的DAG组件的路径正确性进行验算。

14.根据权利要求12所述的装置,其特征在于,所述图形模块还连接有模板模块,用于在已经建立的多个DAG组件形成的连接关系上,框选多个DAG组件,作为一个可调用模板进行保存。

15.根据权利要求12所述的装置,其特征在于,所述图形模块还连接有分析模块,包括:

第一子模块,用于将所述各个DAG组件及连线形成的所述连接关系,分解为数据结构;

第二子模块,用于将分解后的数据结构,按照语句的构成,拆分成多个逻辑算子;

第三子模块,用于所述拆分后的多个逻辑算子,生成相应的语句代码。

16.根据权利要求15所述的装置,其特征在于,所述分析模块还连接有转换模块,用于接收切换指令,对所述分析过程,执行逆操作,形成所述DAG组件图。

一种采用图形化的开发的方法、介质、设备和装置

技术领域

[0001] 本发明涉及计算机程序开发的技术领域，特别是指一种采用图形化的开发的方法、介质、设备和装置。

背景技术

[0002] SQL即结构化查询语言(Structured Query Language)，是一种数据库查询和程序设计语言，用于存取数据以及查询、更新和管理关系数据库系统；同时也是数据库脚本文件的扩展名。

[0003] 现有技术在使用SQL语句做程序开发时，开发方式通过写SQL语句或在文本中编辑SQL脚本。

[0004] 现有的开发方式，开发者必须熟悉SQL的具体语法。对首次接触SQL语言的开发者来讲，学习并熟悉使用一门新的SQL语法都有一定的学习和时间成本。从而影响了开发效率。

发明内容

[0005] 本发明所要解决的技术问题是上述采用命令行或文本方式编辑SQL语句，需要掌握语法内容，影响了程序开发效率。

[0006] 本发明解决上述技术问题，本发明的实施例提供一种采用图形化的开发的方法，包括：

[0007] 将实现组件功能的算子、允许连接的其他DAG组件名称封装在每个DAG组件内；

[0008] 向用户展示可视化的多个DAG组件；

[0009] 接收用户选择多个DAG组件、以及按照每个DAG组件的功能和允许连接的组件来建立各个DAG组件的连接关系，生成DAG组件图。

[0010] 优选地，所述封装的内容还包括：一个或多个的参数或属性配置项；

[0011] 所述接收用户选择的所述DAG组件的过程，还包括：

[0012] 对所述用户选择的DAG组件，显示参数或属性配置面板；

[0013] 接收用户输入的配置的参数或属性数据，并实时校验是否正确。

[0014] 优选地，还包括：

[0015] 在所述连接关系上，每增加一个DAG组件，所述连接关系发生变化后，对新增加的DAG组件的前路的正确性进行验算；所述前路包括连接关系上的各个DAG组件及形成路径；

[0016] 或，点击任一个DAG组件，对其前路的DAG组件的路径正确性进行验算。

[0017] 优选地，所述验算过程包括：检查每个DAG组件的一个或多个的输入、输出的字段列表和类型的正确性。

[0018] 优选地，还包括：在已经建立的多个DAG组件形成的连接关系上，框选多个DAG组件，作为一个可调用模板进行保存。

[0019] 优选地，所述生成组件图之后，还包括：分析所述DAG组件图的代码关系。

[0020] 优选地，所述分析的过程包括：

[0021] 将所述各个DAG组件及连线形成的所述连接关系，分解为数据结构；

[0022] 将分解后的数据结构，按照语句的构成，拆分成多个逻辑算子；

[0023] 所述拆分后的多个逻辑算子，生成相应的语句代码。

[0024] 优选地，所述生成相依的语句代码后，还包括：接收切换指令，对所述分析过程，执行逆操作，形成所述DAG组件图。

[0025] 本发明的实施例还提供一种计算机可读存储介质，存储有计算机程序，该程序被处理器执行时，实现上述的方法步骤。

[0026] 本发明的实施例还提供一种计算机设备，安装有上述的计算机存储介质。

[0027] 本发明的实施例还提供一种采用图形化的开发的装置，包括：

[0028] 封装模块，用于将实现组件功能的算子、允许连接的其他DAG组件名称封装在每个DAG组件内；

[0029] 展示模块，用于向用户展示可视化的多个DAG组件；

[0030] 图形模块，用于接收用户选择多个DAG组件、以及按照每个DAG组件的功能和允许连接的组件来建立各个DAG组件的连接关系，生成DAG组件图。

[0031] 优选地，所述封装的内容还包括：一个或多个的参数或属性配置项；

[0032] 所述图形模块，还连接有参数校验模块，用于对所述用户选择的DAG组件，显示参数或属性配置面板；

[0033] 接收用户输入的配置的参数或属性数据，并实时校验是否正确。

[0034] 优选地，所述图形模块，还连接有验算模块，用于在所述连接关系上，每增加一个DAG组件，所述连接关系发生变化后，对新增加的DAG组件的前路的正确性进行验算；所述前路包括连接关系上的各个DAG组件及形成路径；

[0035] 或，点击任一个DAG组件，对其前路的DAG组件的路径正确性进行验算。

[0036] 优选地，所述图形模块还连接有模板模块，用于在已经建立的多个DAG组件形成的连接关系上，框选多个DAG组件，作为一个可调用模板进行保存。

[0037] 优选地，所述图形模块还连接有分析模块，包括：

[0038] 第一子模块，用于将所述各个DAG组件及连线形成的所述连接关系，分解为数据结构；

[0039] 第二子模块，用于将分解后的数据结构，按照语句的构成，拆分成多个逻辑算子；

[0040] 第三子模块，用于所述拆分后的多个逻辑算子，生成相应的语句代码。

[0041] 优选地，所述分析模块还连接有转换模块，用于接收切换指令，对所述分析过程，执行逆操作，形成所述DAG组件图。

[0042] 通过上述的实施例，将用代码实现相应组件功能的算子，封装成DAG组件，以图形化的方式显示软件开发过程，简化了用户的开发流程，用户不比关注每条具体指令的表述，以及语法关系，这样的开发模式将代码的算子抽象、封装成可视化、可配置化的前端组件，更加直观，便于用户理解，各个DAG组件之间的逻辑关联抽象成连线，用户只需要理解DAG组件的处理能力，就可以通过拖拉拽、配置组件的方式完成程序任务开发。即使是无语言基础的新用户也不需要关心底层的语法。

附图说明

[0043] 图1为本发明实施例的方法流程图；

[0044] 图2为本发明实施例中参数配置的界面示意图；

[0045] 图3为本发明实施例中一种DAG组件图；

[0046] 图4为本发明实施例的验算过程的示意图；

[0047] 图5为本发明实施例的DAG组件与SQL代码的层结构转换示意图；

[0048] 图6为本发明实施例的另一种DAG组件图；

[0049] 图7为本发明实施例中的分解后的三个子图；

[0050] 图8为本发明实施例的装置结构的模块框图。

具体实施方式

[0051] 以下结合附图对本发明的原理和特征进行描述，所举实例只用于解释本发明，并非用于限定本发明的范围。

[0052] 下面结合附图详细说明本发明的实施例，参见图1，本发明的实施例包括：

[0053] S11：将实现组件功能的算子、允许连接的其他DAG组件名称封装在每个DAG组件内；

[0054] S12：向用户展示可视化的多个DAG组件；

[0055] S13：接收用户选择多个DAG组件、以及用户按照每个DAG组件的功能和允许连接的组件来建立各个DAG组件的连接关系，生成DAG组件图。用户可以采用拖拽的形式选择一个或多个DAG组件。

[0056] 通过上述的步骤，将用代码实现相应组件功能的算子，封装成DAG组件，以图形化的方式显示软件开发过程，简化了用户的开发流程，用户不比关注每条具体指令的表述，以及语法关系，这样的开发模式将代码的算子抽象、封装成可视化、可配置化的前端组件，更加直观，便于用户理解，各个DAG组件之间的逻辑关联抽象成连线，用户只需要理解DAG组件的处理能力，就可以通过拖拉拽、配置组件的方式完成程序任务开发。即使是新用户也不需要关心底层的语法。

[0057] DAG开发模式更符合用户的思维习惯。在DAG模式开发，用户可以根据思维习惯，顺序编排组件以及其间的关系，实现表达一个程序任务的处理过程，需要列举一个和业务逻辑相关的实例。而DAG组件图实际上也反映了数据的处理流向。对复杂的数据处理过程，通过托拉拽、配置化的编辑方式也能让其表达过程更加直观。同时基于DAG的编辑模式，也能有效降低后续的开发维护成本。可以更快速的搭建软件的流程框架，从而有效提高了效率。

[0058] 本发明的方法可以适用多种语言的开发程序，例如SQL语言、Scala语言等，每种语言有各自支持的算子，可以用代码描述算子，一个或多个算子可以封装后，实现组件的功能。对于流计算、图计算的方式，都可以按照这种方案实现。在本发明的实施例中，以SQL语言为例进行说明。

[0059] 优选地，在上述实施例中，所述封装的内容还包括：一个或多个的参数或属性配置项；

[0060] 所述接收用户选择的所述DAG组件的过程，还包括：

[0061] 对所述用户选择的DAG组件，显示参数或属性配置面板；

[0062] 接收用户输入的配置的参数或属性数据，并实时校验是否正确。

[0063] 每个DAG组件的功能，都设置有可以配置的参数范围、或属性配置范围等，如果用户配置的数值超过范围值，通过数值比较后，会向用户弹出问题弹窗。如图2所示，相对于现有的方案，需要代码全部完成，在后期的编译过程中，才会看到配置错误的情况，本实施例在软件开发过程中，就可发现配置错误内容，提高了代码的正确率。

[0064] 优选地，还包括：

[0065] 在所述连接关系上，每增加一个DAG组件，所述DAG组件的连接关系发生变化，对新增加的DAG组件的前路的正确性进行验算；所述前路包括连接关系上的各个DAG组件及形成路径。如图3所示，在TT组件的基础上，用户拖拽了组件GroupBy，并采用连线连接，经过连接后，会及时检测GroupBy组件节点之前的TT，形成的路径，是否正确，可采用schema的推算的方式进行验证。这里的TT组件和GroupBy组件形成的连接也可以是一张子图，形成一个单独的SQL语句。

[0066] 通过上述的实时检查，DAG可视化组件规范了算子的使用方式，比如组件的上下游，可以接什么组件，属性配置项允许配置什么样的内容，字段可以用那些字段等，都被严格的规范起来，降低了处理逻辑出错的可能性，即使用户配置出现错误，也能够实时提示出来。便于用户在每次DAG组件连接过程中，实时检测连接的组件以及连接关系是否正确。

[0067] 优选地，还可以通过用户点击某个节点，如用户点击其中一个select节点，系统会验算该节点之前的路径，TT-GroupBy-filter-select的正确性，将这个路径及节点作为一个任务，对形成的路径及节点是否正确进行验算，采用如schema的推算的方式进行验证这个路径及节点的正确性。从而提升了实施的调试能力。

[0068] 优选地，在采用schema实时验算的过程中，在组件间有连线动作，或者组件配置完成、修改时、触发以下流程：参见图4，包括：

[0069] S21：自顶向下检查每个组件的连线正确性；

[0070] 211检查每个组件下游允许连接的组件类型(属于组件隐性配置)；

[0071] 212检查不通过，提示连线错误，本次检查终止；

[0072] S22、检查每个组件参数配置正确性；

[0073] 221检查组件的配置参数是否符合参数规则，比如是否参数类型(整形、字符串)，是否允许为空等等；

[0074] 222检查配置项、配置表达式中引用的字段必须来自于上游组件的输出字段。如：上游源表组件输出两个字段aa、bb，则下游的Select组件配置的表达式中不能引用名字非aa、bb的字段；

[0075] S23、计算每个组件的输出schema；

[0076] 231当连线、参数配置检查通过后进行输出shcema的推算；

[0077] 232 schema主要包括组件的输出字段列表以及字段类型(如：aa String，bb String)；

[0078] 233上游组件输出的schema即为下游组件输入的schema；

[0079] 234每种组件都有固定的shcema推算逻辑，不同组件的解析方法可能不同。比如Select组件通过解析参数表达式获取输出字段：上游输入的schema为a String，b String，Select组件的selectFields配置为aa as a，max(bb)as b，通过表达式解析(可用SQL解析

工具)解析到输出的schema为:aa String,bb String;对于Join组件:假设上游两路输入中,一路输出的schema为aa String,bb String,另外一路的输入为cc String,dd String,则Join组件的输入schema为上游两路schema的合集:aa String,bb String,cc String,dd String。通过这样的验算,可以得出整个SQL的逻辑是否正确。当每个DAG组件节点的检查没有通过,则终止DAG组件图的检查,输出节点配置错误信息。从而提升了代码的纠错能力。

[0080] 优选地,为便于用户后续开发软件更便捷,提升效率,在已经建立的多个DAG组件形成的连接关系上,可以框选已经形成连接关系的多个部分DAG组件,将框选的多个DAG组件作为一个模板保存,后续再次编辑时,可以直接拖拽这个模板,加入到整个开发的软件中。模板内容包括组件的一些公共配置属性(如表的连接信息、表的字段信息、表达式信息等)以及组件间的连线等。从而形成对代码的快速复用,提升软壳的开发效率。

[0081] 优选地,还包括,对DAG形成的组件关系图进行分析,形成最终用于编译的代码。上述的分析过程,是将DAG组件图的组件,分解成最终的代码,如SQL的DAG组件最终分解为可执行的SQL代码。下面详细说明一个具体分解过程的实例。

[0082] 参见图5,在实施例中,主要包括渲染后的形成连接关系的前端的DAG组件,位于图中的第一行。

[0083] 在分解过程中,从第一行的DAG各个组件,向下分解。第一行的各个DAG组件,其形成连接关系的视图如图6所示,这些DAG组件图,进行分解后,形成节点列表和边列表,构成一个json数据体,json为JavaScript对象表示法语法的子集。每一个节点存储了组件类型以及用户在该组件配置面板配置的组件参数,边数据存储了组件间的有向连线的关系。如图4中的第二行的视图层。

[0084] 第二行的视图层接到DAG数据体之后,将DAG子图拆分为多个子图,拆分的目的使每一个子图最终都生成一个单独的SQL语句,拆分的依据为判断每个节点的DAG组件是否需要就此断开,(如本例的create view节点与其后节点需要断开,以形成一个独立的SQL表达语句)。经过拆分后,形成如图5所示的第三行的DAG抽象组件层的内容,即抽象组件层。

[0085] DAG抽象组件层将DAG组件拆分成3个子图,拆分后的三个子图如图6所示。对应图5中第三行中三个虚线框中的内容。在上述每层的传输过程中,每层内容的数据结构也会相应调整。

[0086] 抽象组件层接到视图层的数据后,将内容进一步分解后,得到基础逻辑算子层的内容,如图5中的第四行所示。将DAG组件拆分成一个或者多个基础的SQL算子结构(根据DAG组件封装的功能),如图5或图6中GroupBy组件,可以拆分成一个GroupBy算子和一个Select算子。

[0087] 基础逻辑算子层,依次遍历抽象组件层解析、拆分后的逻辑算子子图,生成对应的SQL代码。

[0088] 如GroupBy组件,可以拆分成一个GroupBy算子和一个Select算子,生成的语句为:CREATE VIEW view_4cc AS SELECT aa,max(bb)AS bb FROM AA GROUP BY aa;

[0089] GroupBy组件对应的子图为图7中左侧的子图1,中间为子图2、右侧为子图4。图5中的基础逻辑算子层中,中间的组件对应图7中的子图2,右侧的组件对应图5中的子图3。

[0090] 基础逻辑算子层,将子图2生成的SQL和子图3生成的SQL分别如下:

[0091] 子图2生成的SQL语句代码:

[0092] SELECT aa,bb FROM view_4cc WHERE aa＞0 and aa＜100;

[0093] 子图3生成的SQL语句代码：

[0094] SELECT aa,bb FROM view_4cc WHERE aa＞100

[0095] 通过上述的步骤,实现了将DAG图代码话,便于工程人员进一步调试,编译代码。

[0096] 在上述过程中,如果用户需要从代码状态切换到DAG组件的视图模式,可以通过切换按钮,将从代码行的窗口切换为DAG视图模式,此时,按照上述代码分析过程的逆运算,转换为DAG视图。以SQL语言为例,具体过程如下：

[0097] 执行SQL切换到DAG的动作触发以下流程：

[0098] 基础逻辑算子层,接收到SQL层编辑的SQL文本内容,利用SQL解析工具将SQL文本内容解析成AST语法树,遍历、解析AST语法树生成定义的基础算子DAG数据结构。其中遍历AST语法树可以使用现有的SQL工具,按照最终的封装的功能,将每一条单独的SQL语句解析为一个独立的逻辑算子DAG结构；

[0099] 抽象组件层,根据抽象组件封装需要,将一个或者多个逻辑算子封装成抽象组件,最终输出抽象组件DAG数据结构;如本示例中:基于通常的语法特征,groupBy算子一定结合select算子使用,因为可以合并为GroupBy抽象组件。本例中最终输出三个独立的逻辑组件DAG结构,如图5所示的上虚线框中组件。

[0100] DAG视图层,将三个独立的逻辑组件DAG结构合并为一个独立的DAG结构。合并的原则是寻找子图之间的输入输出关联。比如图6、图7中的CreateView节点最终输出一个名字为view_4cc的view视图,子图2、子图3的首节点,从名字为view_4cc的源上读取数据,由此可以判定子图2、子图3与子图1之前有数据关联,为子图1的下游子图。合并三个子图形成一个完整的DAG数据结构,如图5中的完整DAG视图层中的图结构。封装的组件,可以按照不同的组件功能封装,如按照数据处理或数据查询分类、按照数据库的定义/计算引擎的对象封装、或数据库的管控权限。

[0101] 前端,接收到完整的DAG图数据结构,渲染为可视化的DAG编辑图,形成如图6所示的形式。

[0102] 优选地,本发明的实施例提供一种计算机可读存储介质,上述实施例中的步骤,可以采用程序代码的形式存储在计算机介质中,该程序被处理器执行时,实现上述实施例中的方法。

[0103] 优选地,本发明的实施例提供一种计算机设备,安装有上述的计算机存储介质。如用于提供下载上述代码软件的服务器,或读取上述介质中的电脑。

[0104] 优选地,本发明的实施例还提供一种采用图形化的开发的装置,包括：

[0105] 封装模块,用于将实现组件功能的算子、允许连接的其他DAG组件名称封装在每个DAG组件内；

[0106] 展示模块,用于向用户展示可视化的多个DAG组件；

[0107] 图形模块,用于接收用户选择多个DAG组件、以及用户按照每个DAG组件的功能和允许连接的组件来建立各个DAG组件的连接关系,生成DAG组件图。

[0108] 优选地,所述封装的内容还包括:一个或多个的参数或属性配置项；

[0109] 所述图形模块,还连接有参数校验模块,用于对所述用户选择的DAG组件,显示参数或属性配置面板；

[0110] 接收用户输入的配置的参数或属性数据，并实时校验是否正确。

[0111] 优选地，所述图形模块，还连接有验算模块，用于在所述连接关系上，每增加一个DAG组件，所述连接关系发生变化后，对新增加的DAG组件的前路的正确性进行验算；所述前路包括连接关系上的各个DAG组件及形成路径；

[0112] 或，点击任一个DAG组件，对其前路的DAG组件的路径正确性进行验算。

[0113] 优选地，所述图形模块还连接有模板模块，用于在已经建立的多个DAG组件形成的连接关系上，框选多个DAG组件，作为一个可调用模板进行保存。

[0114] 优选地，所述图形模块还连接有分析模块，包括：

[0115] 第一子模块，用于将所述各个DAG组件及连线形成的所述连接关系，分解为数据结构；

[0116] 第二子模块，用于将分解后的数据结构，按照语句的构成，拆分成多个逻辑算子；

[0117] 第三子模块，用于所述拆分后的多个逻辑算子，生成相应的语句代码。

[0118] 优选地，所述分析模块还连接有转换模块，用于接收切换指令，对所述分析过程，执行逆操作，形成所述DAG组件图。

[0119] 上述实施例中的装置，可集成在各种计算机设备内，或存储介质内，实现实施例中的各个方法步骤。

[0120] 本领域普通技术人员可以意识到，结合本文中所公开的实施例描述的各示例的单元及算法步骤，能够以电子硬件、计算机软件或者二者的结合来实现，为了清楚地说明硬件和软件的可互换性，在上述说明中已经按照功能一般性地描述了各示例的组成及步骤。这些功能究竟以硬件还是软件方式来执行，取决于技术方案的特定应用和设计约束条件。专业技术人员可以对每个特定的应用来使用不同方法来实现所描述的功能，但是这种实现不应认为超出本发明的范围。

[0121] 所属领域的技术人员可以清楚地了解到，为了描述的方便和简洁，上述描述的系统、装置和单元的具体工作过程，可以参考前述方法实施例中的对应过程，在此不再赘述。

[0122] 另外，在本发明各个实施例中的各功能单元可以集成在一个处理单元中，也可以是各个单元单独物理存在，也可以是两个或两个以上单元集成在一个单元中。上述集成的单元既可以采用硬件的形式实现，也可以采用软件功能单元的形式实现。

[0123] 集成的单元如果以软件功能单元的形式实现并作为独立的产品销售或使用时，可以存储在一个计算机可读取存储介质中。基于这样的理解，本发明的技术方案本质上或者说对现有技术做出贡献的部分，或者该技术方案的全部或部分可以以软件产品的形式体现出来，该计算机软件产品存储在一个存储介质中，包括若干指令用以使得一台计算机设备(可以是个人计算机，服务器，或者网络设备等)执行本发明各个实施例方法的全部或部分步骤。而前述的存储介质包括：U盘、移动硬盘、只读存储器(ROM，Read-Only Memory)、随机存取存储器(RAM，Random Access Memory)、磁碟或者光盘等各种可以存储程序代码的介质。

[0124] 以上，仅为本发明的具体实施方式，但本发明的保护范围并不局限于此，任何熟悉本技术领域的技术人员在本发明揭露的技术范围内，可轻易想到各种等效的修改或替换，这些修改或替换都应涵盖在本发明的保护范围之内。因此，本发明的保护范围应以权利要求的保护范围为准。

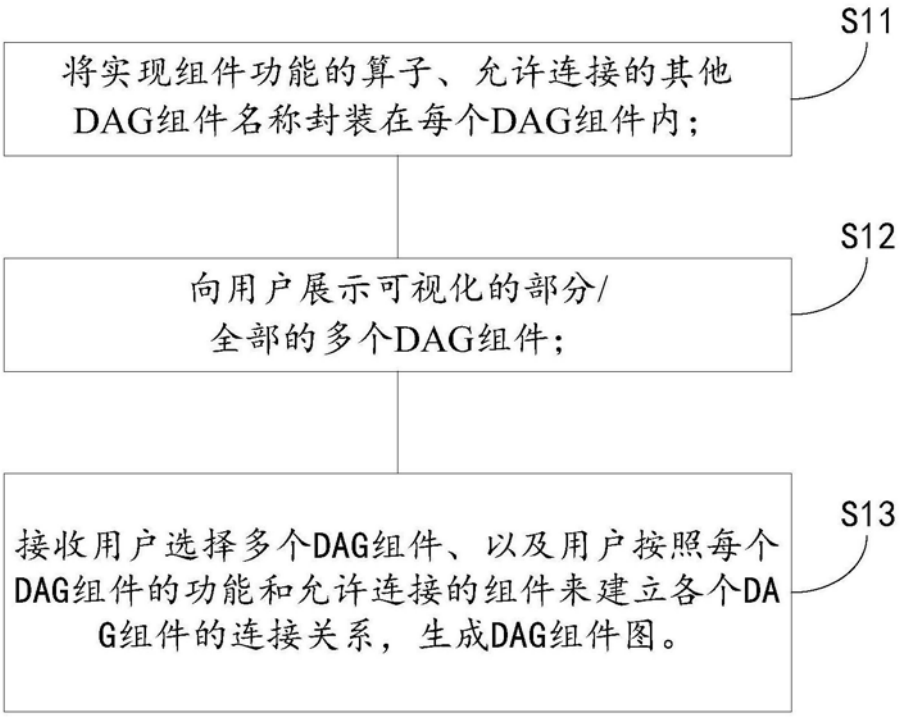

S11
将实现组件功能的算子、允许连接的其他DAG组件名称封装在每个DAG组件内；
S12
向用户展示可视化的部分/全部的多个DAG组件；
S13
接收用户选择多个DAG组件、以及用户按照每个DAG组件的功能和允许连接的组件来建立各个DAG组件的连接关系，生成DAG组件图。

图1

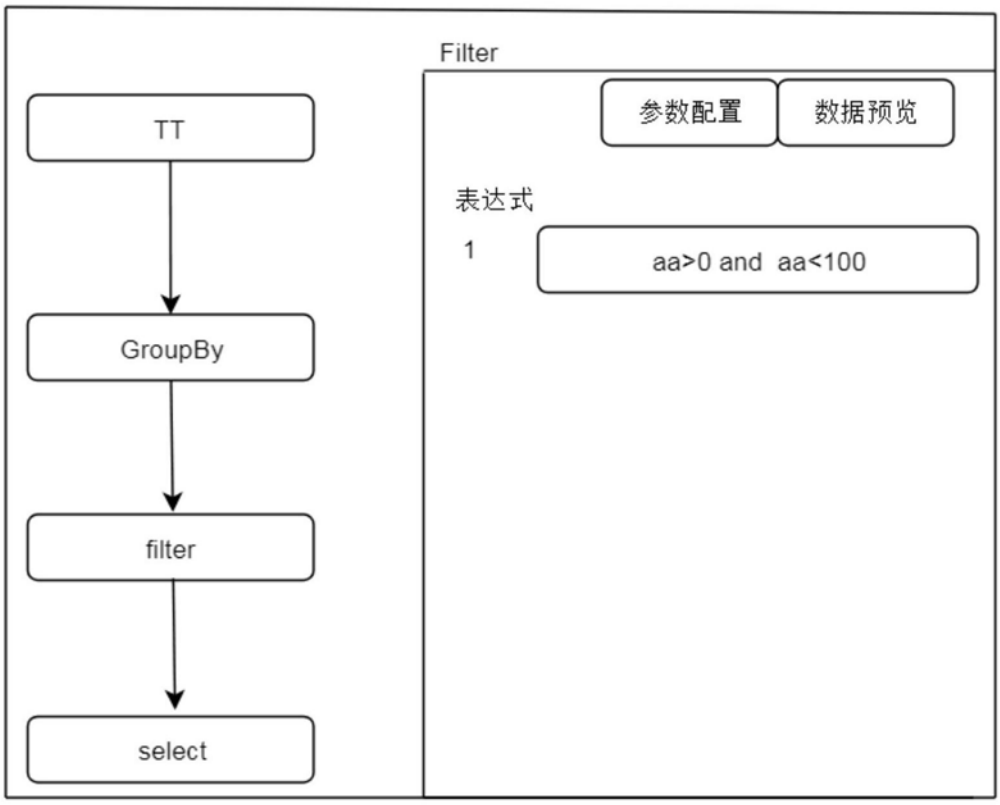

TT
GroupBy
filter
select
Filter
参数配置
数据预览
表达式
1
aa>0 and aa<100

图2

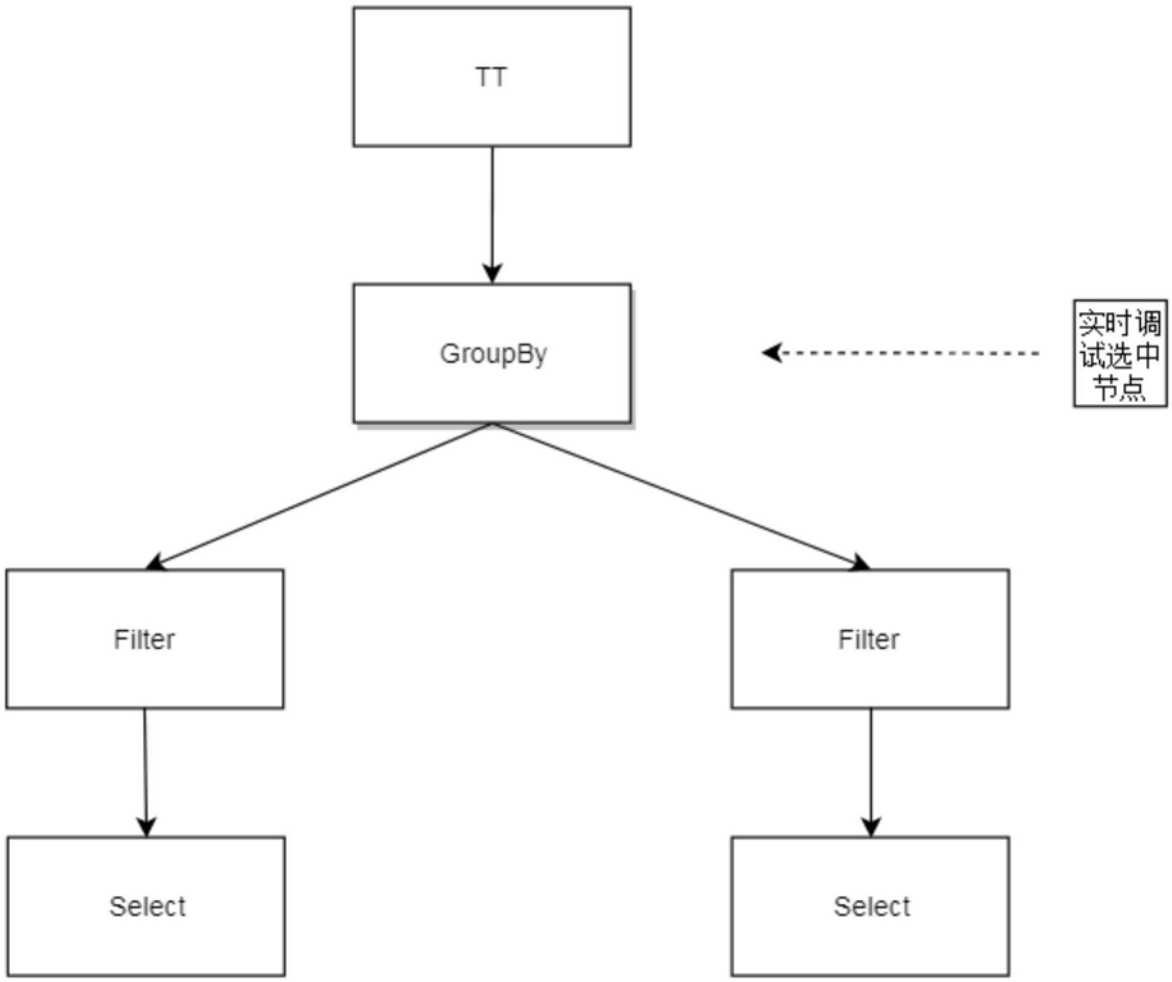

TT
GroupBy
实时调试选中节点
Filter
Filter
Select
Select

图3

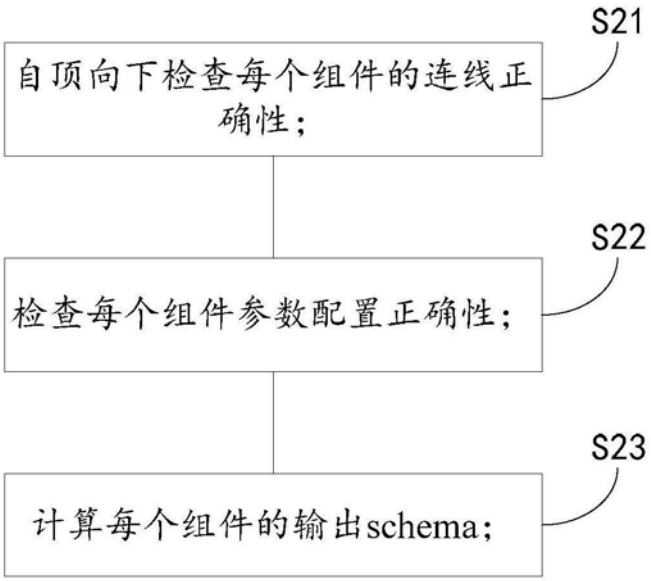

S21
自顶向下检查每个组件的连线正确性；
S22
检查每个组件参数配置正确性；
S23
计算每个组件的输出schema；

图4

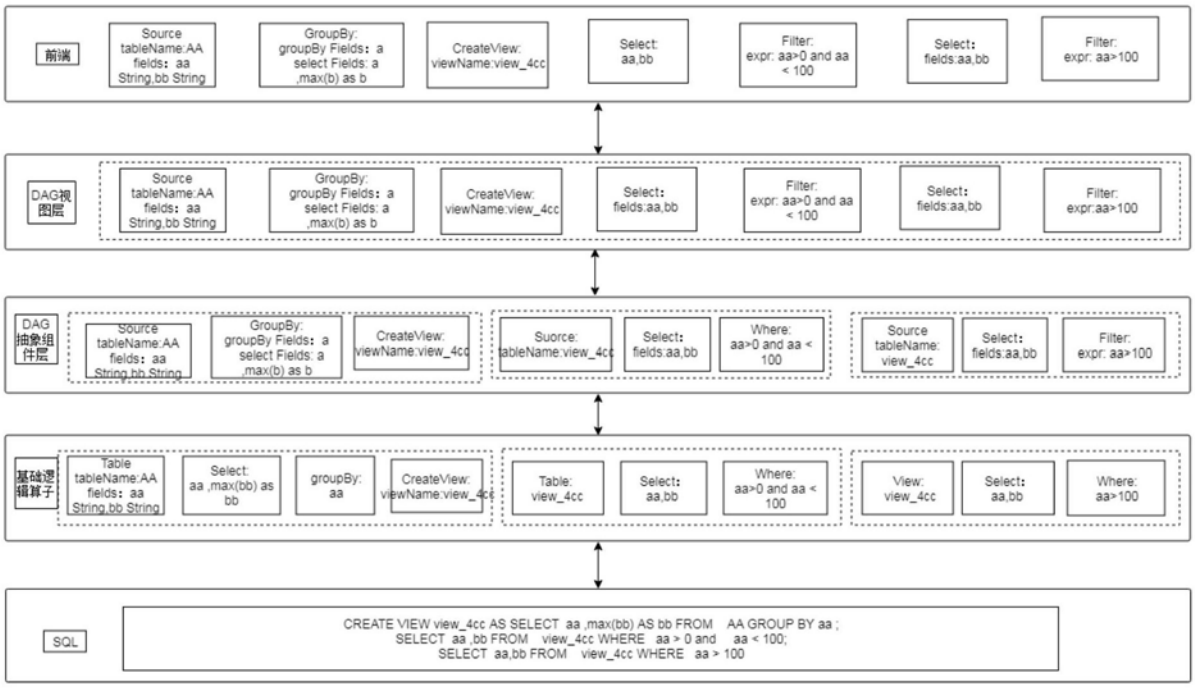

前端
Source tableName:AA fields: aa String,bb String
GroupBy: groupBy Fields: a select Fields: a ,max(b) as b
CreateView: viewName:view_4cc
Select: aa,bb
Filter: expr: aa>0 and aa < 100
Select: fields:aa,bb
Filter: expr: aa>100
DAG视图层
Source tableName:AA fields: aa String,bb String
GroupBy: groupBy Fields: a select Fields: a ,max(b) as b
CreateView: viewName:view_4cc
Select: fields:aa,bb
Filter: expr: aa>0 and aa < 100
Select: fields:aa,bb
Filter: expr:aa>100
DAG抽象组件层
Source tableName:AA fields: aa String,bb String
GroupBy: groupBy Fields: a select Fields: a ,max(b) as b
CreateView: viewName:view_4cc
Suorce: tableName:view_4cc
Select: fields:aa,bb
Where: aa>0 and aa < 100
Source tableName: view_4cc
Select: fields:aa,bb
Filter: expr: aa>100
基础逻辑算子
Table tableName:AA fields: aa String,bb String
Select: aa ,max(bb) as bb
groupBy: aa
CreateView: viewName:view_4cc
Table: view_4cc
Select: aa,bb
Where: aa>0 and aa < 100
View: view_4cc
Select: aa,bb
Where: aa>100
SQL
CREATE VIEW view_4cc AS SELECT aa ,max(bb) AS bb FROM AA GROUP BY aa ;
SELECT aa ,bb FROM view_4cc WHERE aa > 0 and aa < 100;
SELECT aa,bb FROM view_4cc WHERE aa > 100

图5

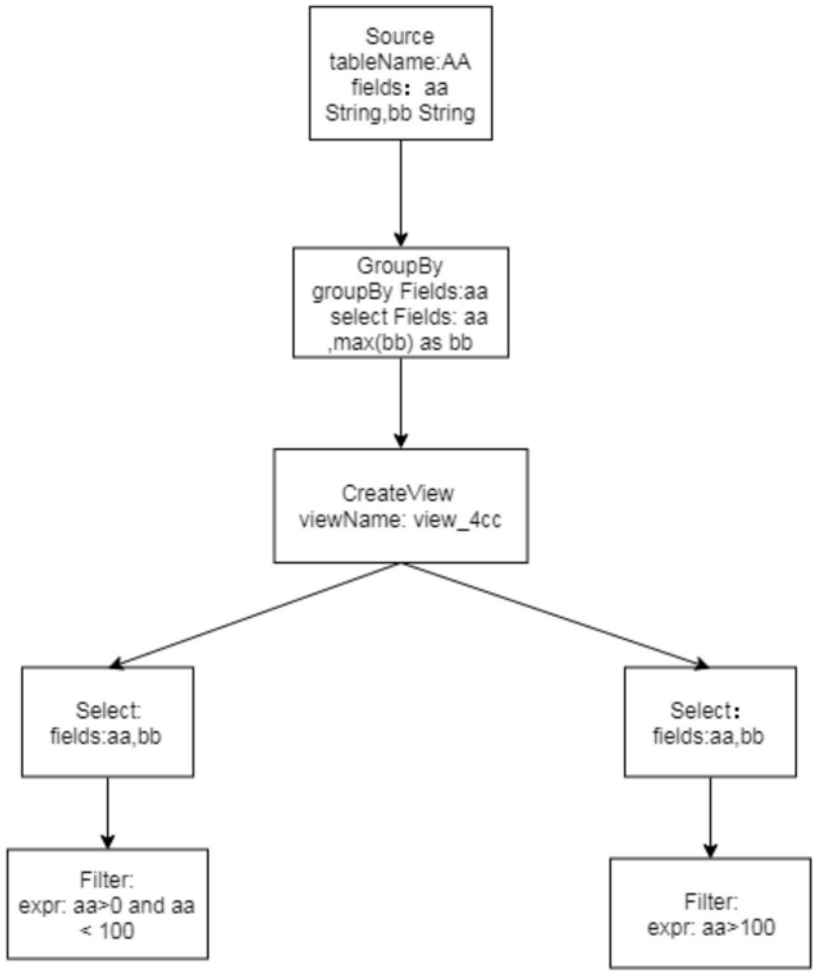

Source
tableName:AA
fields: aa
String,bb String
GroupBy
groupBy Fields:aa
select Fields: aa
,max(bb) as bb
CreateView
viewName: view_4cc
Select:
fields:aa,bb
Select:
fields:aa,bb
Filter:
expr: aa>0 and aa
< 100
Filter:
expr: aa>100

图6

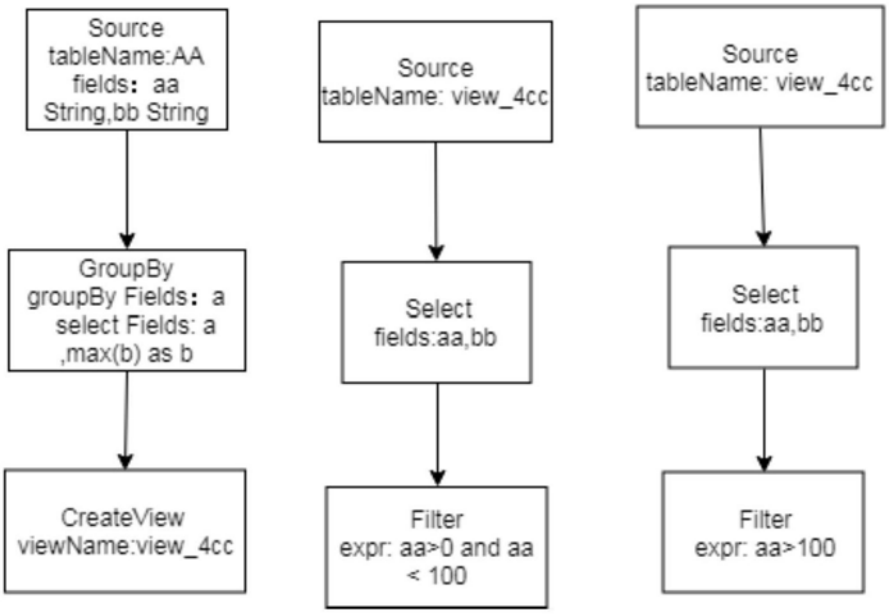

Source
tableName:AA
fields: aa
String,bb String
GroupBy
groupBy Fields: a
select Fields: a
,max(b) as b
CreateView
viewName:view_4cc
Source
tableName: view_4cc
Select
fields:aa,bb
Filter
expr: aa>0 and aa
< 100
Source
tableName: view_4cc
Select
fields:aa,bb
Filter
expr: aa>100

图7

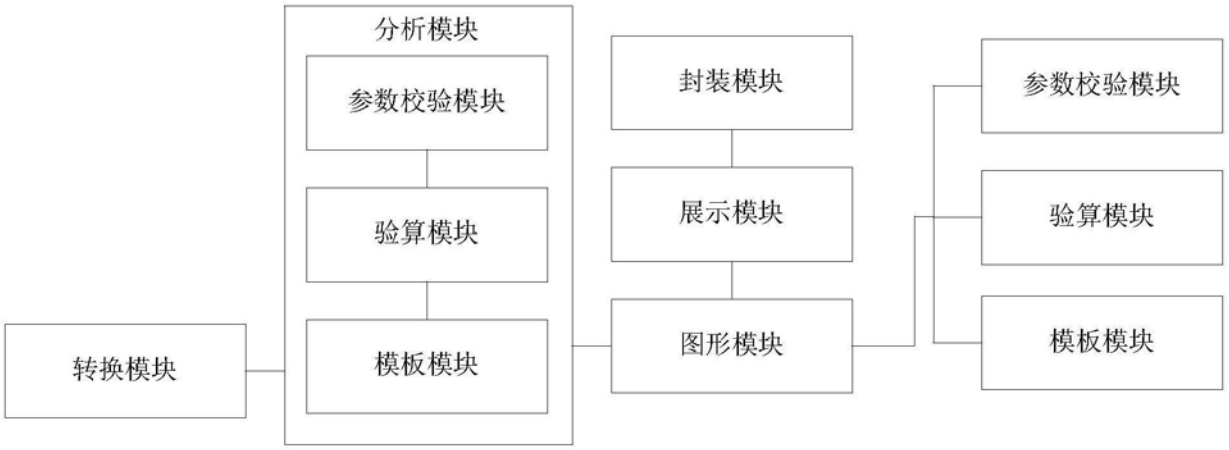


图8